Tracker: The future is present
Carlos Garnacho, Sam Thursfield

Carlos Garnacho: Are we ready?

>> Hello. Okay. It looks like we are ready. Oh, yeah. So, we are continuing track 1 with Carlos Garnacho and Sam Thursfield. With Tracker: The future is present. Please.

Sam Thursfield: Okay. I'm gonna start. So, well Tracker, we know what it is. It's a search service for applications which is used in GNOME. Next slide, please. We're gonna start with a quick history lesson. Very, very quick. So, Tracker is an old project now. It started in 2005. It's not the first full text indexer. I'm not going to talk about the pre-Tracker days. There's an interesting history if you want to do some research.

But when Tracker started, it had several goals. One goal was to provide full text search by default in GNOME. Well, that goal was achieved. Actually, it can do a lot more than just full text search. There's a very flexible database where you can query a lot of things. But full text search is maybe the headline. It was a goal to be fast. And mostly it is. It's acceptable to certain books, we try to fix them as much as we can. There was a goal to be free. And, of course, it's free software.

There was a goal to be private. And privacy in 2005 means your data never leaves your computer. So, Tracker maintained a central search index and it lived in your home directory and that's fine. Everything is private. Privacy in 2020 is slightly more complicated because we now encourage users to install untrusted applications via Flatpak. So, we are more at risk when we do that of leaking data.

And the main motivation for the Tracker 3 version break was to try and make the database, the index, more resilient. So, we have decentralized it so that we can give untrusted apps much less action. They can still do full text search. But maybe only GNOME can only search through your music and nothing else. Okay. That ends the history lesson. Next slide. This is what we will talk about.

We're gonna start with when can we have it? Then we're gonna go through, what's new in Tracker 3. And at the end, of course, we'll quickly talk about how you can contribute to the project. As a spoiler, there is a lot you can contribute. So, next slide.

Okay. This is -- what's the name of your cat, Carlos?

Carlos Garnacho: There is Lewis.

Sam Thursfield: Lewis wants to know, which can we have Tracker 3? Let's see on the next slide. So, the good news is the core work is mostly done. Most of the core code changes were in progress already. Last year when we decided to do a major version break. There are lots of big changes. The database format is different and incompatible with previous versions. The C libraries are

incompatible, the DBus interfaces are incompatible. So, there was no way we can pretend Tracker 2 and Tracker 3 are the same. Any way we can pretend GTK3 and GTK4 are the psalm. Tracker 4 is a major version break. It's parallel installer with Tracker 2. So, you can install it. But there's a catch, if you have Tracker 2 and 3 enabled, they will both crawl the same files so you will have double resource consumption. Which you don't want. So, you can parallel install them. But we want to make sure we don't have both running. And that means really we don't want to have both installed except for developer use cases.

So, it's parallel installable for testing and for us for convenience of developers. But we expect, and we hope users can migrate 2 to 3 with no intermediate stages. Next slide.

Our plan is to release for GNOME 3.38, which is the next major release. And GNOME releases are planned on the Wiki. The dates for 3.38 mean that in one month, more or less, when we enter string trees, if we want to have most of the core functionality ready and tested. And the harder deadline is the 5th of September. It's the last to release. And after that, can't change any code. It's purely fixing critical bug. So, we have about a month in which to carry out our plan. Next slide. Let's see the plan.

So, like I said, most of the core work in Tracker is done. There's only one major feature which we still want to do which is related with on-demand indexing. So, there's one more chunk of work we need to look at. But that is in-progress. Most of the work remaining is in the applications which use Tracker. So, we made a list on the Wiki. I'll paste the link in the chat afterwards because there is an interesting page for everyone to follow the progress.

And here we listed every GNOME core app that uses Tracker. And every library, core library, and also apps in the wider world that also uses Tracker. And they have a tick if the port is merged. And we link to the merge request if that's merge request. So, as you can see, very little is merged yet. But branches are ready for almost every project. In fact, of the core, the only one we don't have is libfolks, which is actually unused. We want to pull that, we don't think it's a priority. We think nobody is using it. Nobody is using the Tracker backend. They're using the evolution backend. There's a branch, but it's not public yet. everything else has branches. What we don't have is review and testing of all these branches. So, we really, really hope that everyone watching can look at these branches.

Two of the apps -- I'll tell you which afterwards, two of the apps you can simply download a Flatpak and test it on your machine in a few clicks. And the others you can review the branches. Even if you don't maintain the app, you can look through and go, I think this is crazy. I think this is good. Everything -- every comment to say this works or doesn't work is useful to us. So, we hope that we can get all of this done in the next month. And if we do, great. We can release everything for GNOME 3.38. GNOME 3.38 depends on Tracker 3. Tracker 2 is dead. Everyone is happy.

Next slide. This is what we want to avoid. This is the works on my machine certificate from the Coding Horror Blog. So, GNOME doesn't have a formal QA team. I can test on my laptop. Carlos can test on his laptop. So, it works on our machines. But we rely on you to make sure that it actually works in every situation. So, we want to move beyond the works on my machine certificate. And next slide.

We have a plan B. You're thinking now, what if not every branch gets reviewed and merged in time? After all, a lot of maintainers are volunteers. Maybe they don't have time to review these

branches. So, our plan B is that the core of GNOME would still depend on Tracker 2 in the next release. So, that would be annoying because we'd have to ship GTK with the Tracker 2 code still. And probably Nautilus as well. Apps could still be in Tracker 3 because of the magic of Flatpak. They will work in the Flatpak bundler even without Tracker 3 on the host. We have been testing this already. It works well.

So, even if we have to delay Tracker 2 for the core of GNOME, apps will still be able to benefit from all the new features by shipping Tracker 3 inside the bundle. And we -- anyway, we recommend that we'll ship Tracker 3 inside the Flatpak bundle. Because otherwise they won't work on older distros. So, your app will continue to work even up to 18-04 which is never going to get it in the base operating system. It can work in the Flatpak. The cost is higher resource consumption. But the app doesn't break. I think it's a fair tradeoff. Next slide, please.

So, a quick note for distribution packages. Hopefully lots are listening and keen to package Tracker 3. So, there are two options. In the plan A, I described where everything is ready for the next GNOME release. You could keep the same Tracker packages. Upgrade to the next version of Tracker and simply say that it conflicts with the older versions. In the case where we have to fall back to plan B, and we have the core depending on GNOME 2 and the apps depending on Tracker 3, in that case, it's better if a distro would ship Tracker 2 for the GNOME core to use and a version of Tracker 3 which is disabled but default but can be activated by the apps. So, that will be a -- kind of a -- a work around for one release cycle of GNOME if we have to do it. So, we can release Tracker 3 and we can get some testing. But it will be disabled by default. We'd avoid the problem of double resource consumption except when users are using.

For example, GNOME Music, which is Tracker 3. And in that case, the music folder could start to be doubly indexed. But that's the price to pay. And it would only affect folders that app is interested in. So, our recommendation, I suppose, is to wait and see. But we may in the end have to ship Tracker 3 in parallel to Tracker 2. It's kind of up to you. So, now I'm gonna hand over to Carlos and he's gonna talk about all of the things which are new in Tracker 3. So, over to you.

Carlos Garnacho: Yep. So, what's new? This is -- by the way. Yeah. So, a big improvement for applications is, of course, data isolation. So, yeah, we'll -- we don't require everything to be shared to everyone. And we allow certain applications to access only the data and access too. So, yeah, this is built on top of SPARQL language features which sits well with SPARQL because while it has notions about graphs and whatnot. So, that they are already set. So, the data which we can rely on and they are coherent by themselves. And they have some nice features that make them quite fit for this isolation. And this is -- and whatnot. And it also sits well with us because we just need a small nudge in the right direction to make use of this and it's -- it depends on the application and the qualities to us. It ranges from trivial to not that much. But, yeah. It's not something that's -- and it relies on reasonably-sized chunks of information. So, it's not too cumbersome to us for permissions to read music or stuff like that.

We have basically graphs that contain all of audio, all of video, all of software and whatnot. So, they are basically the chunks that application can get access to also. But not quite all of it. So here is, for example, the anatomy of a file. Maybe you've -- you know it and whatnot and how it's described. It consists of subject predicated on object. Which they are basically -- the subjects are not yet. They would be like the notes in the graph. And the predicate would be directed to one. So, you say A, B. That would be one set of -- one describing one interaction. And you want more and more and more

and you get a graph of things. So, here is a simplification of the file.

And you can imagine that the graphs problem in every other direction and having different properties and pointing to whether nodes or whatnot and also the video thing. But the idea here is that the file information is split from the content information and the -- what the file contains inside and whatnot. So you have, for example, we have this big green squares which could be the graphs. They basically separate data. And we have a filesystem graph with the filesystem information and you get a file folder relationship or a file size and whatnot. And then you get the video graph. In this case, the file is a video. That will contain all the logical stuff inside the file. The data, or how long it is, or the director and everything.

So, that there's a clear split between the logical data and the physical data which is the file itself.

And how do you do this separation? This is pretty -- pretty easy on the top once you prepare your -- to this separation. If we go back in Tracker 2, this could be a single block. A file and video would be a single element and they would point to the folder and then to the director. And, yeah, well, it's -- it was quite harder to split. This is using a double movement as it should be used. It's an improvement. And it also sits well with us, as I said.

So, this is the actual stuff that makes an application talk only to one graph or the other. We -- we specify one policy which consists of the DBus name for the files. And we tell it that it can access a specific graph. So, this is -- I guess this stands -- let's see if screensharing works...

Sam Thursfield: I should note this metadata is set at the time the Flatpak is built. So, the app declares up front, you should only have access to music. When the user installs it, verify it. It's a music player, that's fine. Whereas if the app declares it wants access to everything, then the user hopefully will consider before installing it and investigate a bit more to see that it's trust worth why I.

Carlos Garnacho: Just wait a moment. I think Firefox is not liking... I tried with the testing before. Okay. Now -- after I exit and come back? Or should we go ahead?

Sam Thursfield: Maybe we should go ahead. We can show the demo at the end, right?

Carlos Garnacho: Yeah. Let's see. Hopefully.

Sam Thursfield: The demo is effectively an app being unable to access something.

Carlos Garnacho: Yeah. Yeah. It's basically that. I was just going to show that the app actually works. Yeah. So, yeah. The next point would be stores and endpoints. So, yeah, this is -- the actual -- the actual most successful change in the Tracker API and structure. So, before you had an installer which would be an entity you don't know where it is. And the Tracker connection gets, and you get a single object and you talk to something and everyone talks to the same thing.

So, yeah. Well, this doesn't pan out, obviously. We tried to assimilate everything. So, we now have a division between stores and endpoints. Which could be, the store, what that is. The tracker store. And create one with tracker_SPARQL_connection_new. And it creates in the sense that you specify actually a location for that database and it's your database. It's not served with any -- and whatnot.

And if you want to make it public, you have a Tracker endpoint. Which makes your connection public in whatever tunnel you specifically. And DBus currently, the spec that finds HTTP endpoints and whatnot. We are not there yet. The document we are missing to implement. But almost there.

Now we have the install which is local and the endpoint which is an option to not make it public. And then what do we need? We need to query that database. So, and actually the SPARQL language ties it together with service graph patterns. This is a function line for the syntax that allows you to query different services instead of your own database. And it's -- yeah. Pretty simple. It's -- you do wrap with service, DBus, something. And what you used to do inside, you do it on the outside service.

And actually, with this mixed together, we get a distributed database where everyone gets to own they layout. They get to choose what they -- what access they allow to everyone else and everyone can be on that actually.

And with this we have like three sample, three model applications. And we have, for example, application A which would be the simplest one. Like consuming tracker-miners data. We want to look for files for -- yeah. Well, the user would be a proper example of this. It only wants data, it doesn't want to change anything. So, there's this construct, tracker_SPARQL_connection_bus_new. Which it basically works like TrackerSparqlConnection get from Tracker 2. But you have to have an explicit name to this tracker. That means that, yeah, you have to talk explicitly with one service or another. In most cases it would be tracker maintainers. Well, we hope that will change. That's important.

We have application model B. Which it does -- well, this sample, this application will be in the notes. Because it does -- doesn't care about Tracker-miner's data. But it does use Tracker for its own data. In that case, you have tracker_SPARQL_context_new. This works in that you can create your own database. And just go with it.

And the application C is a mixture of both. So, it actually gets to consume tracker-miners data. But it also wants to store its own data. An example of this would be with collections or music with playlists or favorite -- favorite songs and whatnot. So, here the approach would be to create a tracker_SPARQL_connection_new, which is pretty much like the previous one. But then you have to use explicit service graph patterns with SSL. These are the bits of syntax that allow you to create other services. The neat thing with this is that, well, you have your own local database and you are allowed to query on it. And you are allowed to update on it. But the service graph patterns, they can only run on queries. So, you can only query other services. But you can not run updates on random services. You can only run updates on your own database. So, the language syntax actually makes the only model that works for us.

Yeah. And as Sam said before, yeah. We have to -- we used to have user data in trackers 2, the database. Yeah, we need to offer some support in order to allow migration of data for whether to consider it, or wanted to run regression scripts. So, it's pretty much everything in place. We already have a tracker export command in Tracker 2.3 meant to help with this. And it's basically helps you migrate application-specific data into your own databases. And I think this is for you, Sam.

Sam Thursfield: Yes. Okay. So, that's about apps. What about users? Well, why aren't users first? Because Tracker is not intended as something end users in general interact with directly. The design

is that applications or the shell will expose search results or maybe a voice assistant like the one Giovanni is talking about right now. Contact Tracker. But in general, if a user notices Tracker, it might be because it's gone wrong. So, what's new for users? Data security is the big thing, as we've already talked about.

One thing which I'm quite happy with, can you show the next slide.

Carlos Garnacho: Yep.

Sam Thursfield: We have some online documentation. The eagle-eyed of you will notice that this is just a GNOME pages turned into a web page. It means we keep the pages up to date and you can read the documentation of the commandline without having to have manpages installed. And we now use GitLab pages to put up a simple website with some instructions for people who want to find out more about the project and want to contribute. An FAQ for people who have seen Tracker on top and want to know what it is. And we're quite happy with it. Can you go to the next slide?

So, talking of the commandline interface, some users will want to interact with Tracker. Perhaps because they have complicated use cases. So, our goal is to work out of the box by default. But because a user's home directory can be enormous, it can be an NFS mount, it can be a 15 terabyte, very slow hard disk, we can't do everything by default. So, the default configuration of Tracker is relatively conservative. It indexes the default xdg content directories and indexes the mom directory, but non-recursively. So, that gets most data that are let's say -- a person who doesn't have huge amounts of files in their home directory might use.

But more advanced users will want to configure Tracker to do more complicated things. The commandline interface is also useful for developers and for bug reporters. It provides a way to see what went wrong. Perhaps if a file isn't indexed and you think it should have been. So, the news is that we have done some work on the commandline tool. It's now more like git in that we can install custom commands from different modules. We want to improve it more. So, for example, Systemd has brought some quite nice improvements to commandline times such as automatic pages. And we want all that.

We want -- at the moment Tracker reports errors into the Journal. We think it would be good to save errors on to the disk and be able to show them in the CLI as well so we don't have to tell users how to and dig in the Journal necessarily to report a bug. And yeah. So, this work is lower priority. Because we don't have to block the release if the CLI isn't ready. So, we have plans. We haven't finished all of this work yet because we are more concerned with making sure that the API changes and the application ports are ready. But we do want to polish the CLI. And you think it's now back to you, Carlos.

Carlos Garnacho: Yep. Yeah. So, for current developers, well, I tried to split this section between current developers and prospective developers. So, yeah. So, what you will get here is a streamlined API. As in, yeah, well, in the Tracker tool we have like three libraries. One the classical SPARQL library, which is the only one staying. And then you have another one to implement miners which no one ever used in more than a decade. And certainly you have the tracker mine per control API which some projects use. But, yeah. Well, it's not -- it doesn't fit the picture as well for Tracker being a core SPARQL thing. So, basically, we've streamlined the API into the SPARQL library which is Tracker. The Tracker git module. And then we have the -- for Tracker miners, the most prominent daemon is the

system miner. And that will be for desktop Tracker 3, miner.files. Give us the interface. That will be the API to talk to it. To set priorities or to us to index things and whatnot. But that's not Tracker -- Tracker's core task anymore.

And we have better documentation. And Sam did some nice work on better documentation and whatnot that could apply for custom orthonogals. And we are now following the tracker -- the SPARQL 1.1 spec by the letter. So, yeah, we have a more constrained vision of what things are so we can document things a lot more better now.

It's standards compliant. So, every SPARQL tree you might see on the Internet, it would work on Tracker. So, that's -- that's a nice improvement. It's got its easy sandboxing. You can now run fake endpoints or run your Tracker miner files for testing with some files and with some nice cover here with improvements that have been in the sandboxing of applications.

So, yeah. It's -- well, it's got easy sandboxing. Which at this point, I will say what I have been talking about before. These three lines in the Flatpak and manifest which help you listen to this portion of information or this other. And it also got better validation tools. If you are a developer or using Tracker, you can -- you have better tools at your disposal. As your disposal, like -- yeah. You have better syntax errors which actually point you to the place where you committed this mistake. You have also a set of common line tools that may help you to -- to actually test your stuff, test your queries or test your own ontology or whatnot. It helps developers better rather than having a single central service.

If you run something to test, you better take care of linting it properly and not do too much. Because you don't to want mess with it. You can create fake endpoints that are accessing memory and everything.

And this actual still fits. Tracker was in devices, or was -- got most of the development in the Nokia days and it was developed to work on the Nokia devices, the very old ones based on Linux. And while it has quite some constraints that are right now like 256 megs of memory and I don't remember the processer. But yeah. Well, the thing is that we have managed to stay quite in the -- in the initial footprint in the memory footprint that was initially in the device for those devices. It's nice that after so many years we have gotten any further. And it's also fast. It could be even more faster for updates. But that's not the most optimized code.

It also is more likely to run because you're not updating -- running updates all the time. You are most likely querying. And it handles 6 million queries in 30 seconds. Everything recording in the circumstances. It's trying to spin out audio bumps and everything. That's, yeah. Nice. It gives a profile of the baseline that Tracker -- of what Tracker can do.

And, yeah. Well, the thing is that for GNOME developers, most of them are using Tracker miners. And they are indexing. So, they are used of the Tracker by trades because they need the index data. But what we actually have here, and we think it's actually a pretty good solid thing for storing your own data. A nice solution for application data. Because it's still local, it's still fast. And you get all the benefits from SQLite. Just you get the different mindset that SPARQL involves. Graphs and notes instead of columns and tables.

You get actually more stuff on top. SPARQL has some nice features. As in a fully introspectable

and whatnot. You can do -- it allows pretty complex things. Even it ranges from the very simple to the very complex. So, yeah, you can do services across different -- queries across different services or do backups or bulk exchanges of information between endpoints or whatnot. It's got some benefits that SQLite doesn't offer. And it's not no impositions.

Tracker used to have -- use a mode. and this is the format of the data and this is how you use it. Not anymore. Let's say this. The thing is that you can actual describe your own data format. an anthology that describes the classes and the classes they have and may have different types or may point to different other classes. So, you basically describe the evaluation of the data in your graph. That's an escape from the world. So, yeah, you can provide your own and use your own data. And your own data is actually yours. It's not in a shell database read about by everyone and whatnot.

And even if you wanted that, you could, for example, create an endpoint for your own application and what not. You can have benefits from sandboxing this infrastructure with them. It's actually -- well, Tracker miner is one version of them. You could be another.

And this is where it gets crazy. As in had, yeah, well, nothing of this is done yet. But it's kind of crazy, yes. But actually where we concentrate right now. On the one hand, your application does not necessarily have to stay within your application. For example, we have a shell search asking for favorite files and whatnot. And that is actually app service query as well. It could be a service query if you were on endpoint. And even if not, if you used Tracker for your own data, like, for example, to use us for the favorite files, you could actually implement a lightweight search provider that just spawns Tracker and no other ancillary stuff. Like get or other stuff. Just for the SaaS providers, that could be as fast as possible.

Another thing that is kind of nice here is that, well, if the -- in talking about introspection of data and about how we have -- you can import and export loads of data between -- between endpoints and whatnot. We have endpoints. So, that you can talk between them. And this is essentially data synchronization. If we have an endpoint which can talk across a local network, we could have data synchronization about the same application and you could do separate machines or things like to. So, yeah, maybe it could help in the -- in the -- well -- it could open the possibilities to a synchronicity by the settings, possibly.

And another possibility is sprawling search. Instead of just searching in your own machine, you could have a set of trusted machines in your local network. And you could be able to query on them if they are turned on. And you could be able to search on them or to look for stuff on them just like if they were your own. Also, with local machines, you could find them locally in your own computer. So, yeah. it can go either way if it's used locally currently.

And I think -- it is yours already.

Sam Thursfield: Okay. Yes. So, last part is how you can contribute. The biggest way you can contribute is to test what we have. So, the easiest thing... there are a couple of apps. I think GNOME Music is one of the closest to being ready. And there's a branch on the GNOME Music GitLab that will give you a Flatpak bundle that you can simply install and see if it works. What it should do is index your music directory and sort your music. There's also a GNOME Photos branch, which is similar. That is less-finished. So, make sure you read the bug report before saying, oh, this doesn't work. Because lots of stuff doesn't work. It gets as far as showing your photos. That's a good step.

There is more to do in porting GNOME Photos. We haven't made bundles that would work on all the systems. But it's something we would like to do. And another thing you can contribute, in fact.

Look at how we did it for Music and Photos and you can do the same in other apps. Let's go to the next slide. So, yes. You can test the apps. There is a slight bug in GitLab at the moment. It's supposed to give you a link in the merge request to the .flatpak artifact. Instead, it links to a TAR file, which is not much use. When you look at the merge request, you have to browse the artifacts and use the one with the .flatpak extension so it works the way we want it to.

If you're interested in deeper testing, I use Fedora myself. So, I've made a Copper repository where you can download my custom packages of Tracker 3. For other distributions, there are equivalents to Copper. PPA's, for example. Unstable repositories, it would be great to see Tracker 3 packaged in these so more and more people could stall it and test it. Developers can review our merge requests against applications. Even if it's not an app you maintain. A comment to say this is crazy, or this is great, is useful.

And if you want to get involved in Tracker yourself, there are plenty of things -- interesting things you could do. Get in touch. We're in IRC/matrix. We read Discourse. And there is also -- for viewers watching in the future -- you can always use the help wanted label on GitHub.GNOME.org. And you can find issues tagged help wanted. Hopefully that will be interesting.

So, I think that is everything. We have time for questions. And maybe -- maybe screensharing will decide to work and we can do the demo. I'm not sure. Well, why don't -- maybe I can answer a question or two while you set up the demo.

>> Yeah. You have 5 minutes for presentation. So, take your time. And you can find the questions in the shared notes. There is a link to the -- to the Etherpad.

Sam Thursfield: Yes. I can see it. The first question, which in fact was answered in the slides. But I'll read it. If any app is mainly distributed as a Flatpak, how can I migrate while not losing the ability to run it on older systems? The answer is to bundle Tracker 3 miners inside the Flatpak. You have to install some custom service files and your application needs to query the host miner FS and then the local miner FS if it doesn't work. It's working in music and photos so you can look there for reference. Good news. We can see the demo after all!

Carlos Garnacho: Yeah. So, this is running a Flatpak on GNOME Music.

Sam Thursfield: Could you make the font size bigger?

Carlos Garnacho: Oh, yeah.

Sam Thursfield: Much better.

Carlos Garnacho: So, this actually works. This is GNOME Music using Tracker 3 on the host. Tracker 3 on my system. But it would work the same if I took Tracker 3 out. So, this is -- takes time to load the covers. But yeah. This is it. And the thing is that, well, if we tried to get in there, it's also shipping the get Tracker line times even though it's not necessarily. But for testing this stuff, it's kind of nice. So, yeah.

So, here I'm querying for MusicPieces. And this is -- are the -- the actual, well, the names for these -- these are all the files. It's actually the audio. If we can grab one of these. Tracker 3. Oh. Yeah. Whatever. But you get this information. It's a song. It's -- but you will get only to see the stuff contained -- extracted from the content of the file. The CODEC, stuff like that. This is -- you don't get to see nothing but the URL. Or the UI of the file. That's all that you see from the file content. And, yeah. For example, let's try to do the same for videos for fun. And we get nothing. Even though if I get into Totem.

This thing... it has written stuff. Which are the actual media files which are inside of my computer. So, this is about it. It's not too flashy. But it shows.

Sam Thursfield: Okay. We still have some time for questions, I think. Now that we've proved that it's not vaporware, it does indeed work. So, the next question, how would distros disable Tracker 3 by default if plan B goes into action? They would only install Tracker 3 if something links against Tracker 3. Would that be sufficient?

No, my suggestion is, Tracker 3 -- well, Tracker is started through XDG Autostart. So, if we wanted to disable it by default, we simply remove the Autostart file. So, you'd have Tracker 2 would have the XDG Autostart file, and Tracker 3 would not have the Autostart file. And only started on a dev OS application when an application needed it. So, it gives us on demand application. And the next question -- oh. Carry on.

Carlos Garnacho: Yeah, well, I was going to reply to that one. Yeah, so, first of all, is indexing faster compared to Tracker 2? Yes and no. Well, the actual -- well, first of all, this talk wasn't about Tracker miners per se. It was about Tracker, the SPARQL thing. It's not about indexing. But, yeah, well, if you ask. The indexing is -- first time indexing is not that much faster. Because that's higher ground. And we also rely on get Trackers, rely on libraries like distro to open file formats and we depend on the actual speed of those libraries to index.

There's some libraries that fare better than others. Basically, there's some files that run slow because of the library is big. Basically. And the second time indexing, catching up with changes or adding the monitors in case of startup where everything is already indexed is actually faster. Because, well, yeah, well, we also spend some time in there. So, thing that's -- we have the opportunity to look closer at the code and it's faster. It's mostly -- also in that regard. As in checking end times for files as fast as possible. So, that's a plus.

Sam Thursfield: I think performance in general, it's interesting -- it's something we're quite interested in. Because obviously if you make a distributed database, the performance characteristics are interesting.

Carlos Garnacho: Yeah. Well, it's got different pitfalls, basically.

Sam Thursfield: Yeah. So, this is one reason we really to want get as much testing as possible so it can make sure we optimize where it's needed.

Carlos Garnacho: Yeah. Yeah. And the next question is data portability. Something I'm very conscious of at the moment is data portability. Either migrating data between local machines or syncing to a cloud service. Could be either by the app or simply saving data to a location that's

synced by Nextcloud/Dropbox, et cetera. That's what I have been talking about this pie in the sky ideas. I mean, we don't really have a distributed model. We are just lucky and get an endpoint implementation that allows us to query between separate machines.

We already have the client side because, well, for example, Wikipedia's endpoint to query Wikipedia. And that works with Tracker, actually. But yeah. Well, we are -- the other -- [lost audio] -- state personal and not get out of your logged in network.

Sam Thursfield: That was a very interesting question. All right. Yeah, oh.

>> Please, finish. You have some time.

Sam Thursfield: Okay. So, I think it's a very interesting question because Tracker could integrate with something like Nextcloud, for example. Transferring files is kind of out of scope, like Carlos said, I think. There are already tools for synchronizing files. But synchronizing data we store in the database like maybe bookmarks, starred files, photo albums, this is where it's more interesting.

>> So, I think that's all of the questions. Thank you, Carlos and Sam, for an excellent presentation. We are going into break until 18: 45. And after that, we will have a keynote with Joshua Simmons. Tune in. But for you, see ya!

Sam Thursfield: Okay. Thanks for watching!